SIFT Report

Junhao Hua [†]

November 22, 2013

1 Introduction

The Scale-Invariant Transform Feature(SIFT) is a method for extracting distinctive invariant features from images that can be used to perform reliable matching between different view of an object or scene[1]. In the past 10 years, the SIFT has been widely used in many application, such as object detection, object classification, stereo correspondence and motion tracking. This report describes an implementation of the SIFT using *MATLAB* mixed with *C/C++*. In the next part,the report presents the main **function** that extracting the SIFT feature and its **data structure** in our program. After that, Some experimental results followed.

2 Main Function: SIFT()

You can using this function to get the **keypoints** and **descriptors** of your input image. There are many sub-functions needed to be detailed described.

2.1 Parameters Setting

Table 1 lists the parameters setting.

2.2 Difference-of-Gaussian function: DiffofGauss()

This function calculated the **Difference of Gaussian**. The input image should be converted to gray and to be normalized. The initial image is **incrementally** convolved with Gaussians to produce images in the scale space. The Gaussian image is down-sampling by a factor of 2, so the image resolution will decrease 1/4 with the increase of octave. We assume that the minimum resolution is 4x4 $pixel^2$. For the continuity of scale space, there are $S+3$ images needed to be produced in each octave. So we can detect local extrema points at S different scale from the $S+2$ DoGs.

2.2.1 Function Interface

- **Input.** The normalized gray image.
- **Output.**The difference-of-Gaussian Matrix(see below)

2.2.2 Data structure

Some important big matrix describe below:

- **Octave{i}(:,:,j)** means the j-th gaussian-smoothed image in i-th octave,$L(x, y, \sigma)$.
- **DoGs{i}(:,:,j)** means the difference of j-th and (j+1)-th Gaussian-smoothed image.

[†]Junhao Hua, Department of Information Science and Electronic Engineering,Zhejiang University, huajh7@gmail.com

Symbol	Code Variable	Descrption
σ_0	sigma0	the amount of prior smoothing, set $\sigma_0 = 1.6$
S	S	the number of intervals of scale space in each octave, set $S = 3$
σ	sigma_pre	a blur of origin image,set $\sigma = 0.5$
k	k	the constant multiplicative factor,$k = 2^{1/S}$
\	OCTAVE_NUM	the number of Octave.
\	SCALE_NUM	the stack size of each Octave, equal to S+3
\	DOGS_NUM	the number of DOGs in each Octave, equal to S+2
\	threshold	a threshold that discard week local extrema point
\	MAGNIF_SIZE	Magnification factor
\	SUBREGIONS_BINS	the number of subregion's row or column
\	ORIENTS_BINS	the number of orientation histogram

Table 1: Parameters setting

2.2.3 Inline function: imGauss2Dsmooth()

This function produced the scale space of an image, $L(x, y, \sigma)$, that is produced from the convolution of a variable-scale Gaussian with an input image. Considering the time efficiency, It was written in C language as it has plenty of non-vector and non-matrix computing. If the function's input and output arguments are well defined, then we can call this function in MATLAB conveniently.

2.3 Local Extrema Detection: findlocalmax()

This function detect the local maxima DoGs in the whole scale space. Local maxima point means that this point should have highest DoGs compared with its eight neighbors in the current images and nine neighbors in the scale above and below. In our implementation, we using a **threshold** to redefine such thing called 'local maxima' for better stability. Namely if A point is higher than B point at some threshold, then we think A is larger than B.

In order to find local minimum points, we could convert the points value to its **opposite** number. Then the minimum problem becomes maximum problem. After we got the index of local extrema points from the output, we need to using ind2sub() to convert subscript values into an array.

2.3.1 Function Interface

- **Input.** Difference-of-Gaussian, threshold.
- **Output.** The index of key points candidate.

2.4 Accurate KeyPoints Localization: keypoint_selection()

This function performs a detailed fit to the nearby data for location, scale, and ratio of principal curvatures.

2.4.1 Discard low contrast points

Using the equation

$$\hat{x} = -\frac{\partial^2 D^{-1}}{\partial^2 x^2}\frac{\partial D}{\partial x} \tag{1}$$

and

$$D(\hat{x}) = D + \frac{1}{2}\frac{\partial D^T}{\partial x}\hat{x} \tag{2}$$

to reject the unstable extrema with low contrast.

In programming aspect,

- it firstly approximated the gradient vector and Hessian matrix using differences of neighboring sample points. Note that equation (1) involves matrix inverse operation which is not a easy thing. In order to avoid it,we turn to use the Gaussian elimination algorithm for solving the linear system.
- If the resulting offset $\hat{x}$ is larger than a threshold suggested 0.5 in Lowe's paper, then the point's location is updated by adding the offset and its value(DoG) is changed using the equation (2).Otherwise, it replaced the current sample point by its neighbor and calculated the offset again. The maximum searching times is set 6.
- After everything is ready, it discarded the points, where its absolute value of DoG is less than a threshold suggested 0.03 by Lowe.

2.4.2 Eliminating Edge Responses

We can calculate the *trace* and *Determine* of Hessian matrix immediately as we have got the all the elements of Hessian matrix in the previous step. Apply the equation

$$\frac{Tr(\mathbf{H})^2}{Det(\mathbf{H})} < \frac{(r+1)^2}{r} \tag{3}$$

where using the valued of $r = 10$, we eliminates keypoints that have a ratio between the principal curvatures greater that 10.

2.4.3 Function Interface

- **Input.** Key Points candidate calculate, DoGs, threshold.
- **Output.** The renew key points.

2.4.4 Data structure

- **KeyPoints_OCT(:,:,:)** is the 3D - Key points in each octave, each dimension represents the points location(x,y) and its scale.

2.4.5 boundary Constraint

Suggestion: Before call this function, it's better to remove points close to the boundary so that the reminding points are in a Gaussian window.

2.5 Orientation assignment: CalcKeypointsOrient()

This function calculates the orientation to each keypoint based on local image properties.

2.5.1 Build the orientation histogram

For each key point, we build its orientation histogram according to the following steps.

- Form its Gaussian-weighted circular window with a σ that is 1.5 times of its scale.
- For every pair points in window, it calculate their gradient magnitude and orientation using pixel differences.
- The orientation is divided into 36 bins. Add the correspond orientation to its histogram bins.
- smoothing the histogram by linear interpolation.

2.5.2 Assign the orientation

The peak that is within 80% of the highest peak is used to also create a keypoints with that orientation. Therefore,there will be multiple keypoints at the same location. Assignment process is according to the below steps.

- Find the highest peak in the histogram.
- Find local peak that's within 80% of the highest peak and also larger than its 2 neighbors.
- Add a Δ to histogram values, where Δ is the result of a parabola interpolation that fit 3 histogram values closest to each peak.

2.5.3 Function Interface

- **Input.** 3-D Key Points, DoGs, S, σ_0
- **Output.** The orient-assigned Key Points. Note that Keypoints has 4 dimensions Now.

2.6 Local Image descriptor: CalcDescriptor()

Previous steps found keypoint locations at particular scales and assigned orientations to them. This ensured invariance to image location, scale and rotation. Now we want to compute a **descriptor** vector for each keypoint such that the descriptor is highly distinctive and partially invariant to the remaining variations such as illumination, 3D viewpoint,etc. The keypoint's descriptor can be calculated by following step.

- Pre-computer the gradient of each pixel in the image of the whole scale space. Then using its gradient calculate its magnitude and orientation value.
- For each keypoint, it processes each pixel in keypoint's Gaussian window and also in the $16*16$ region around the keypoint. It computes the gradient magnitude and orientation value of each pixel.
- Next step, it divided the $16*16$ regions into $4*4$ subregions. And a set of orientation histograms is created on those subregions with 8 bins each. Namely the calculated gradient value in the previous step is assigned to orientation histogram in associated subregions. However, this gradient is weighted by Gaussian factor and $1-d$, where d is the distance of the sample from the central value of the bin.
- the histogram is normalized to L2 unit length in order to enhance invariance to affine changes in illumination.
- Set the values in the unit feature vector larger than 0.2 to be 0.2. And then normalize the vector again.
- Finally, we got 128-dimensional descriptor for each keypoint.

2.6.1 Function Interface

- **Input.** 3-D Key Points, DoGs, S, σ_0
- **Output.** The orient-assigned Key Points. Note that keypoints has 4 dimensions now.

3 Feature Matching: siftmatch()

We've found the feature of a image, represented by keypoints' distinctive descriptor. Then, we're going to match the feature/keypoints of two images. The best candidate match is found by identifying its nearest neighbor. The nearest neighbor is defined as the keypoint with minimum Euclidean distance for the invariant descriptor vector.

3.1 Function Interface

- **Input.** Two set of descriptor
- **Output.** the matched pairs, who's value is the index of its associated keypoints.

4 Plot function

- PlotScaleSpace(): plot the smoothed image in each scale.
- plotKeyPoints() plot the keypoint with the orientation and magnification.
- plotmatchPairs() plot the matched keypoint and its connection of two images.

5 Experiment

We use the dataset from Lowe[3] and Vedaldi [2] to test our code.

5.1 Image Scale Space

Firstly, we plot the Gaussian smoothed image at each scale space. Figure 1 shows an example of a book at difference scale space. It shows that image is getting blurred with the increase of scale.

5.2 Keypoint selection

Next, we're going to check out the effects of keypoint selection on a natural image. Figure 2(b) shows the 1272 keypoints at all detected maxima and minima of the difference-of-Gaussian function, while figure 2(c) shows the 946 keypoints that remain following removal of those with a value of $D(\hat{x})$ less than a threshold. And figure 2(d) show 728 keypoints stayed after eliminate keypoints that have a ratio between the principal curvatures greater than 10.

5.3 Feature Matching

In this section, we use two images showed in figure 3, left one is a single book and right one is the books with extensive background clutter so that detection of objects may not be immediate even for human vision. Figure 4 shows that the keypoints of two images extracted by SIFT algorithm. As we've got the descriptor vector of two images, the nearest neighbors matching algorithm can be immediately applied. Figure 5 shows the final matching result. As we can see,the it is pretty good.

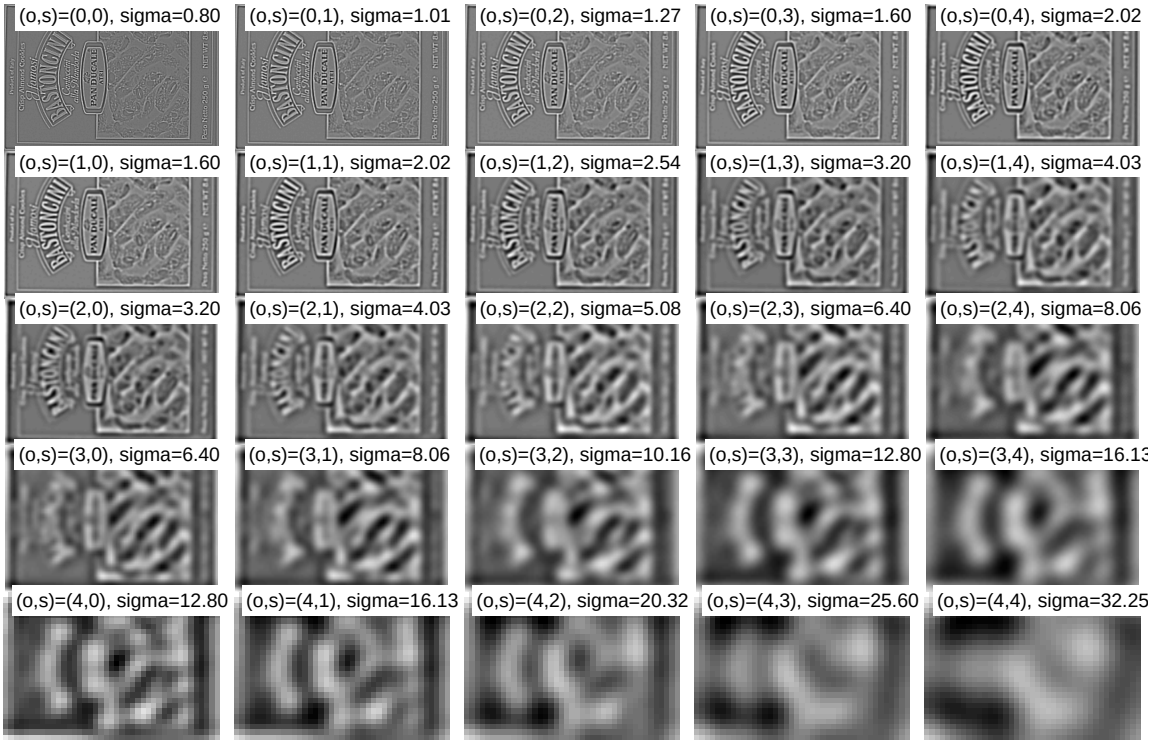

Figure 1: This graph shows the smoothed book image at difference scale. Lower case letter 'o' represents the number of octave layer where the image stayed, and 's' represents the number of stack in this octave where the image stayed, 'sigma' represents the scale of this image.

6 Conclusion

In this report we summary the functions and corresponding parameters and data structure used in the implementation of SIFT algorithm. And experimental results shows that the SIFT keypoints is quite useful due to their distinctiveness, which enables the correct match for a keypoint to be selected from a large database of other keypoints. Although the dimension of the descriptor i.e. 128, seem high, descriptors with lower dimension than this don't perform as well across the range of matching tasks.

References

[1] Lowe, David G. "Distinctive image features from scale-invariant keypoints." International journal of computer vision 60.2 (2004): 91-110.

[2] Andrea Vedaldi. http://www.cs.ubc.ca/ lowe/keypoints/

[3] Lowe, David G. http://www.robots.ox.ac.uk/ vedaldi/code/sift.html

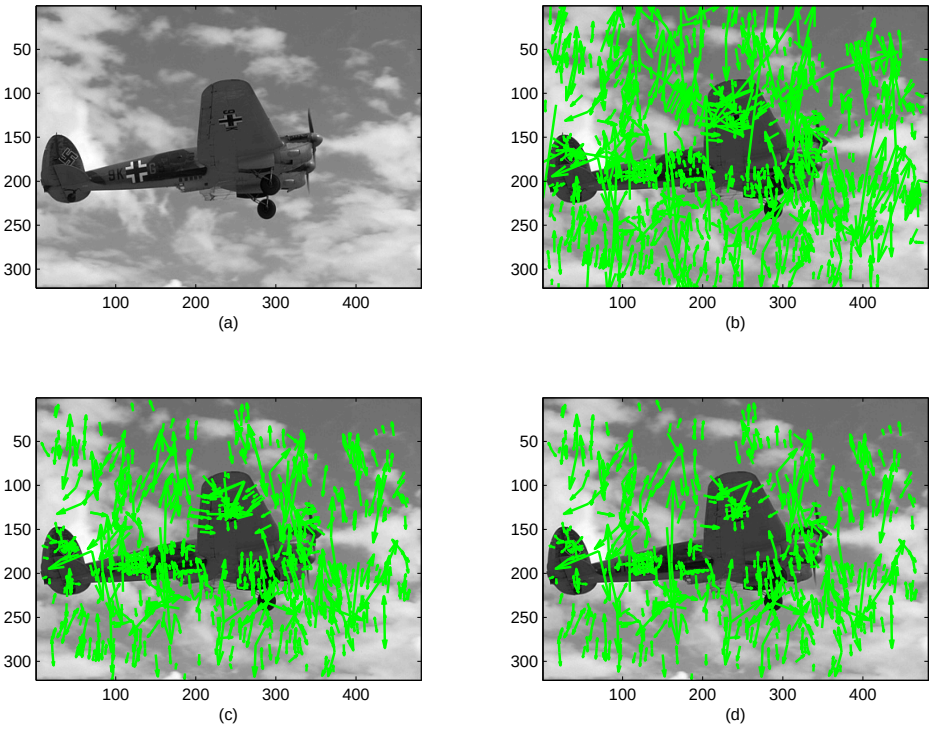

Figure 2: This figure shows the stages of keypoint selection (a)The 481 × 321 pixel original image (b)The initial 1272 keypoints locations at maxima and minima of the difference-of Gaussian function. Keypoints are displayed as vector indicating scale, orientation and location. (c) After applying a threshold on minimum contrast, 946 keypoints remain (d) The final 728 keypoints that remain following an additional thresholds on ratio of principal curvatures

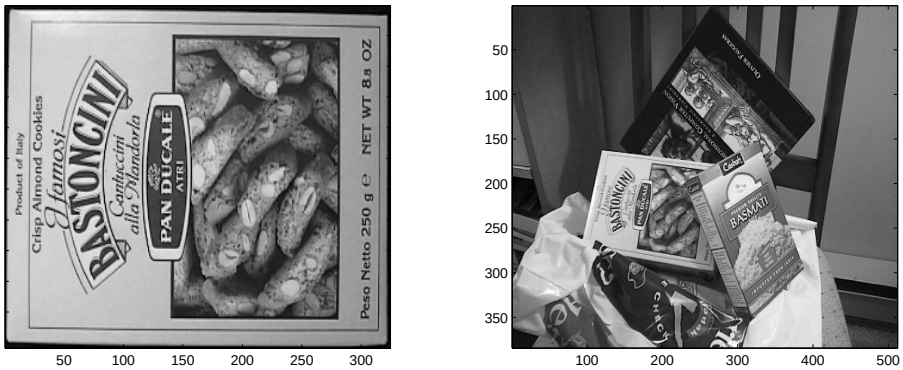

Figure 3: the origin images

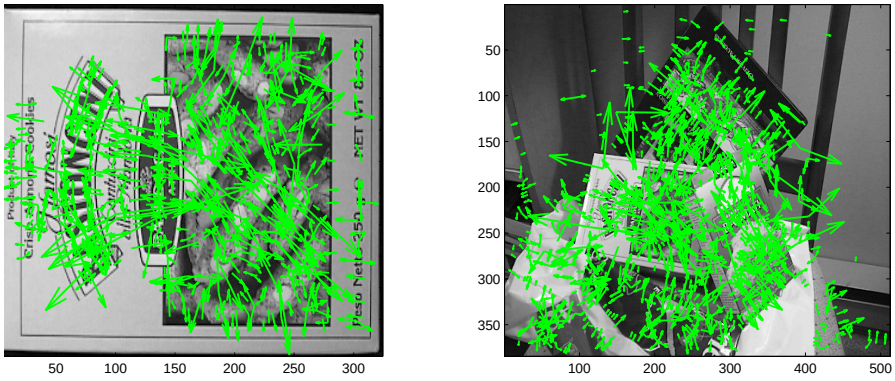

Figure 4: the keypoints of two images

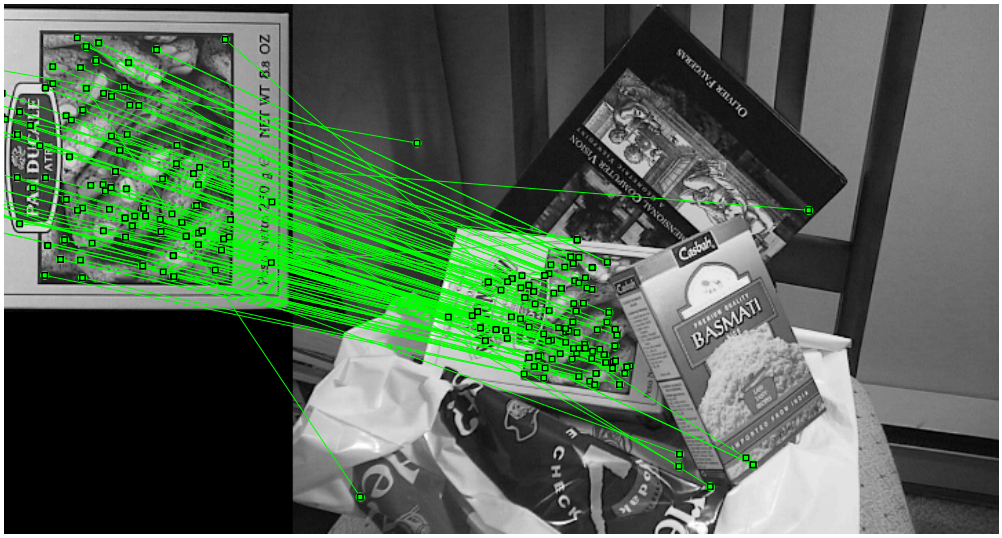

Figure 5: the nearest neighbor matching result using SIFT descriptor vector